

FIG. 1

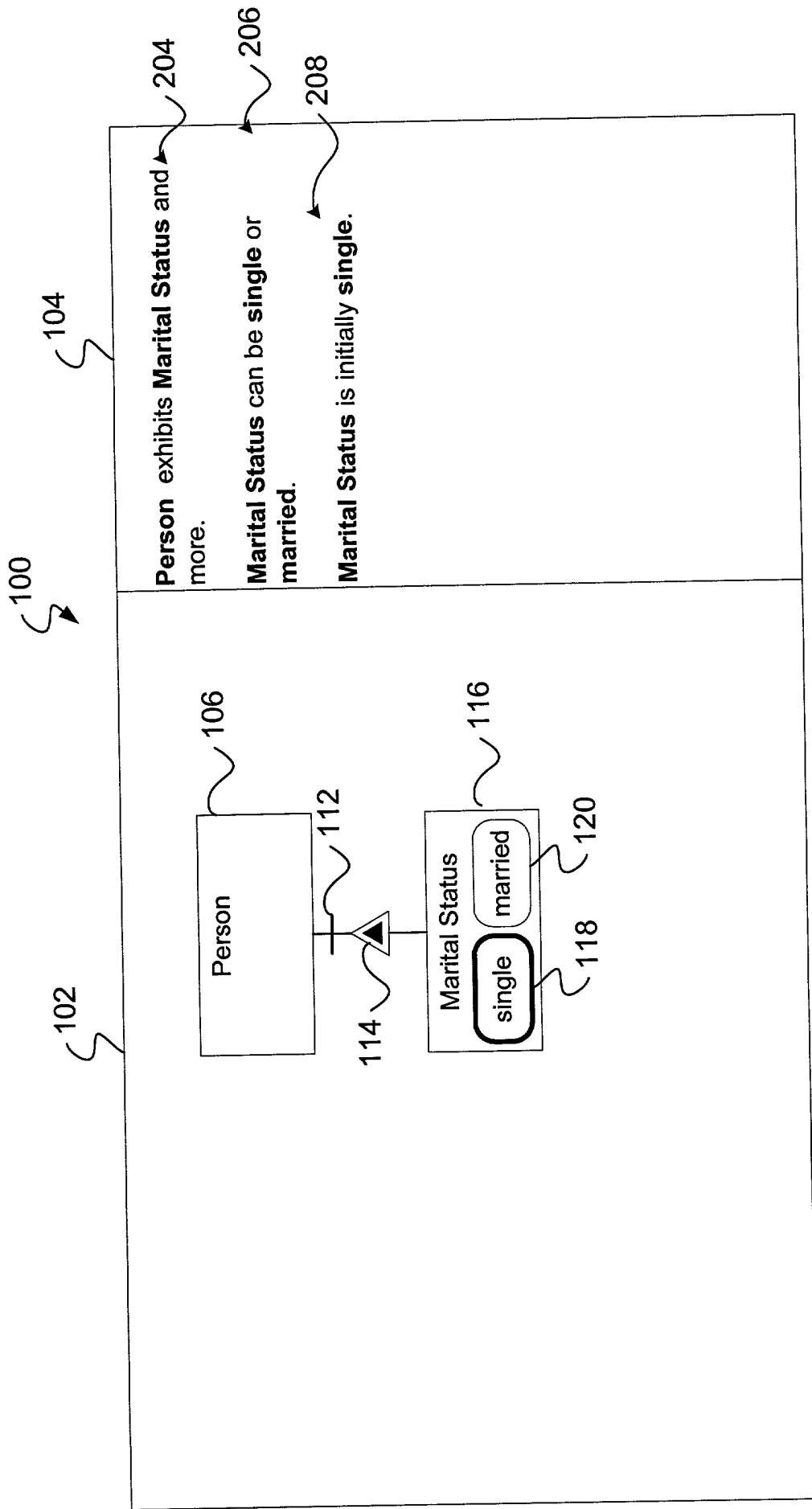


FIG. 2

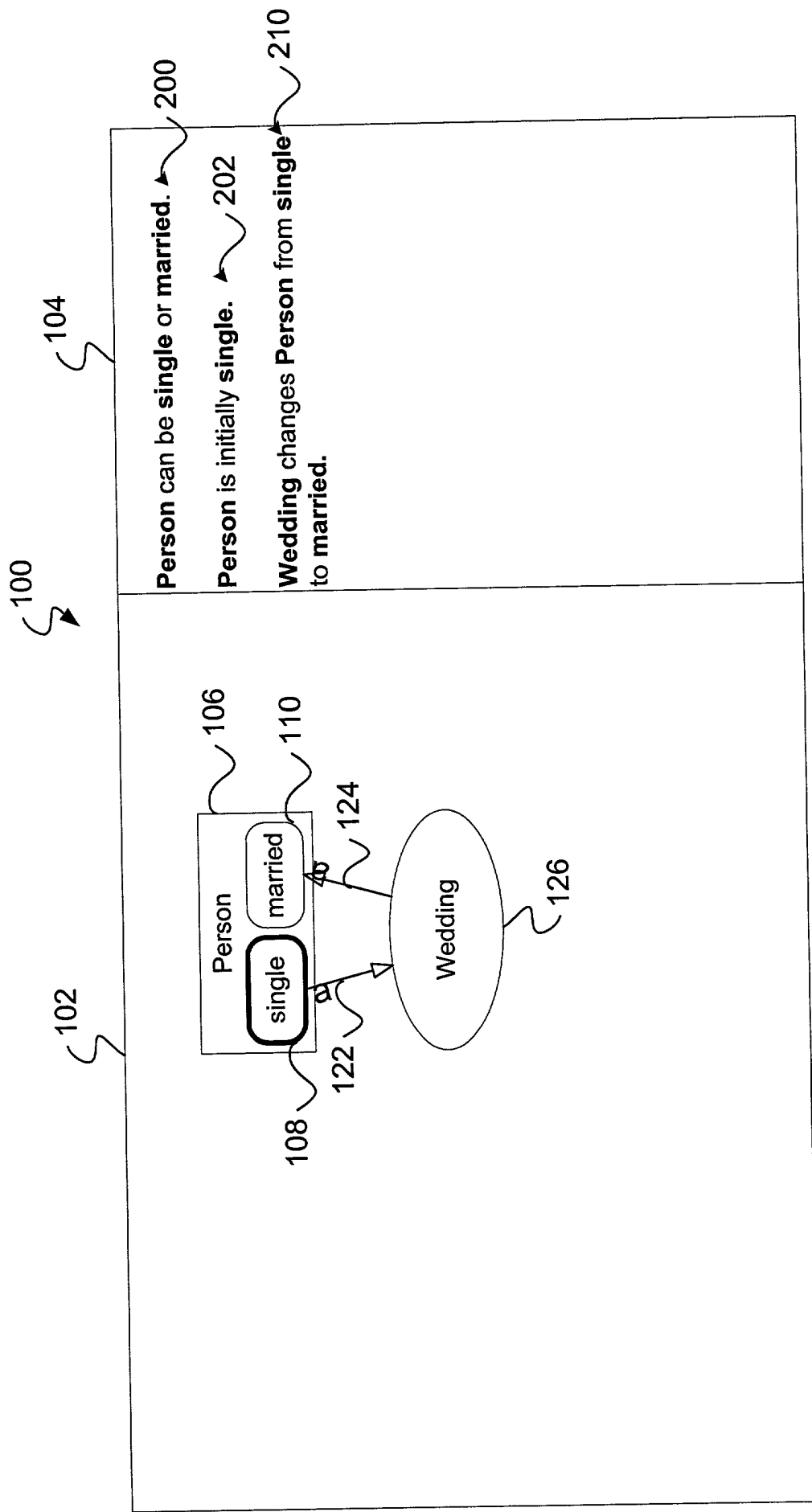


FIG. 3

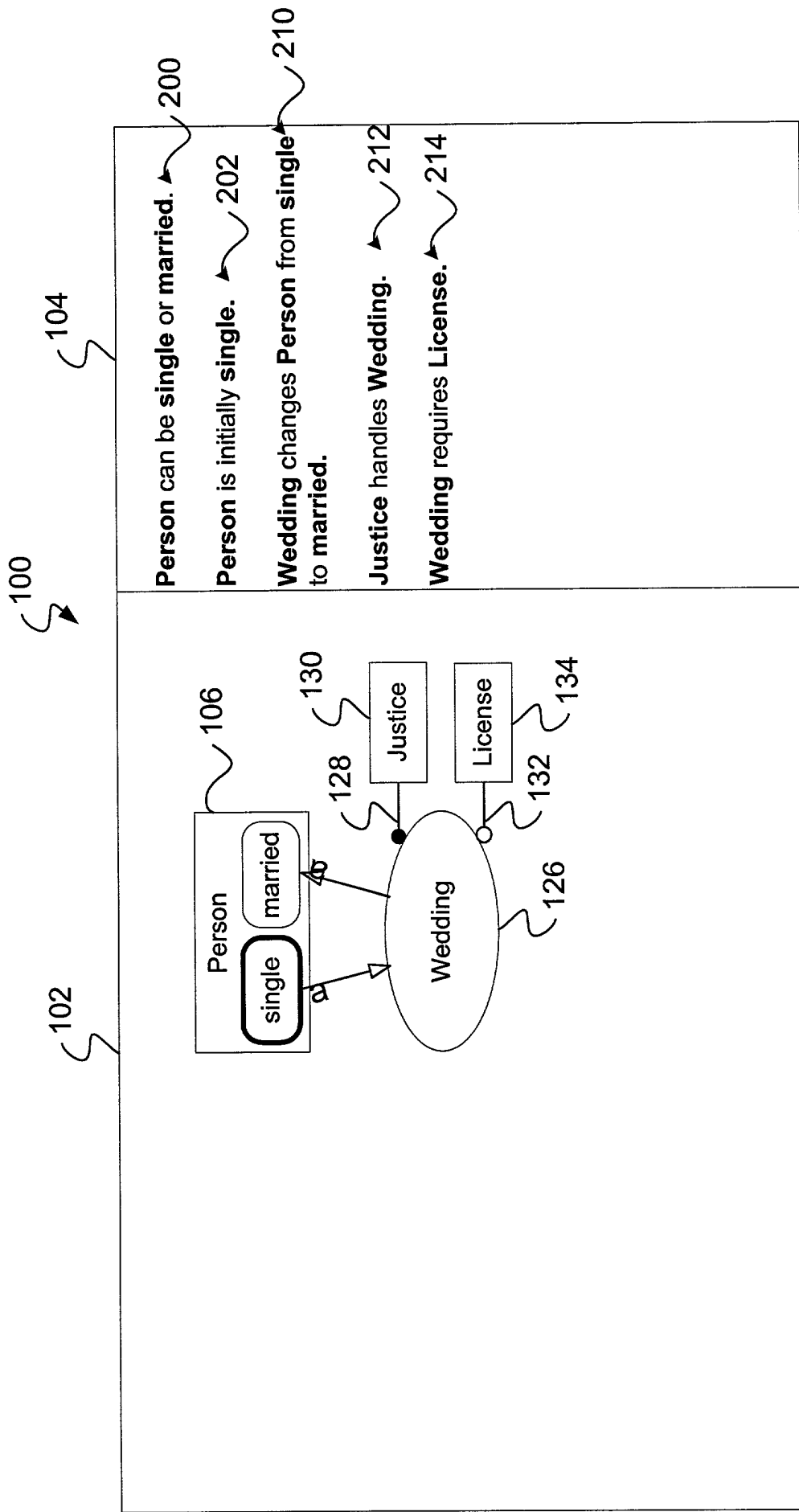


FIG. 4

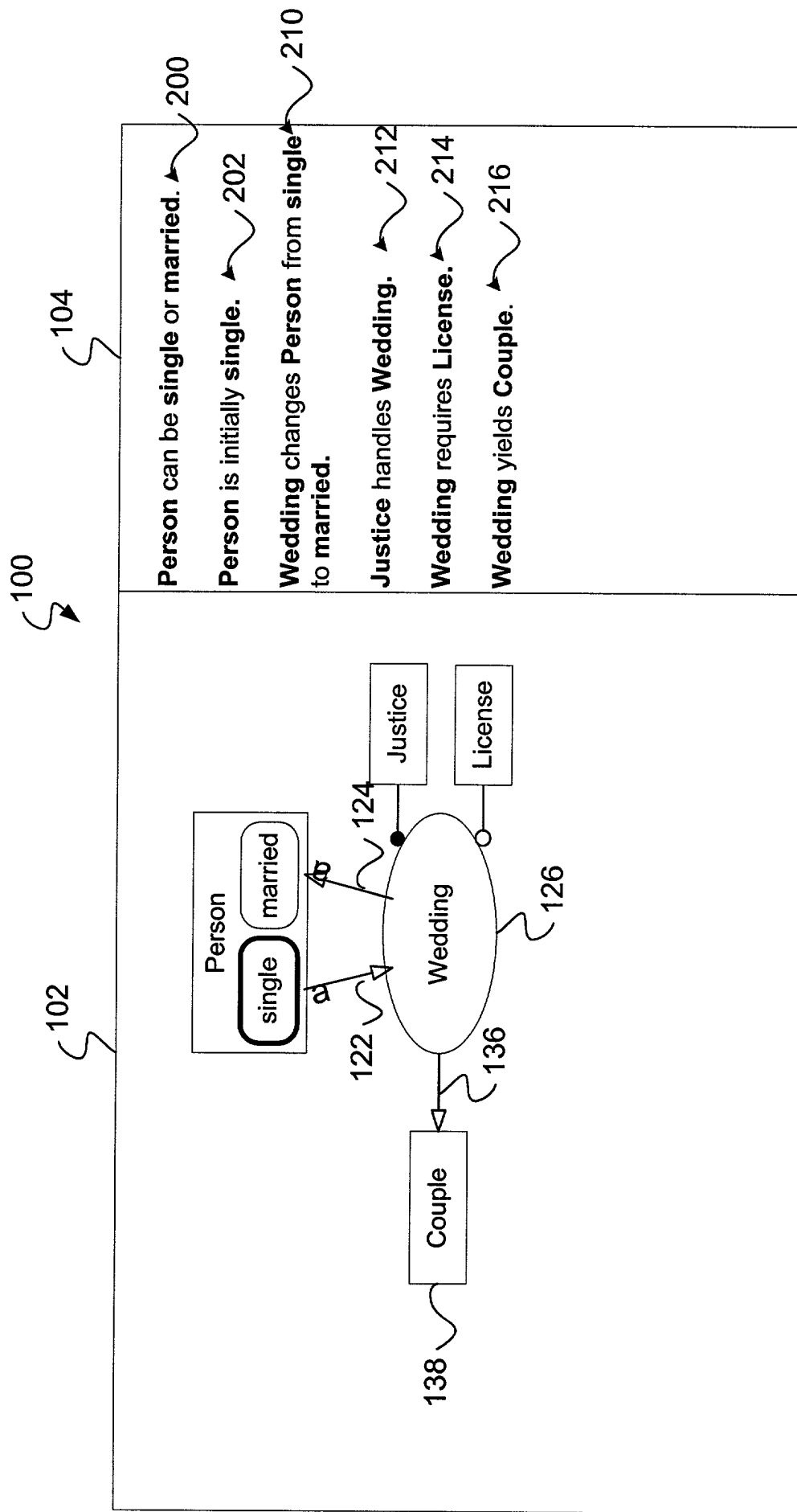


FIG. 5

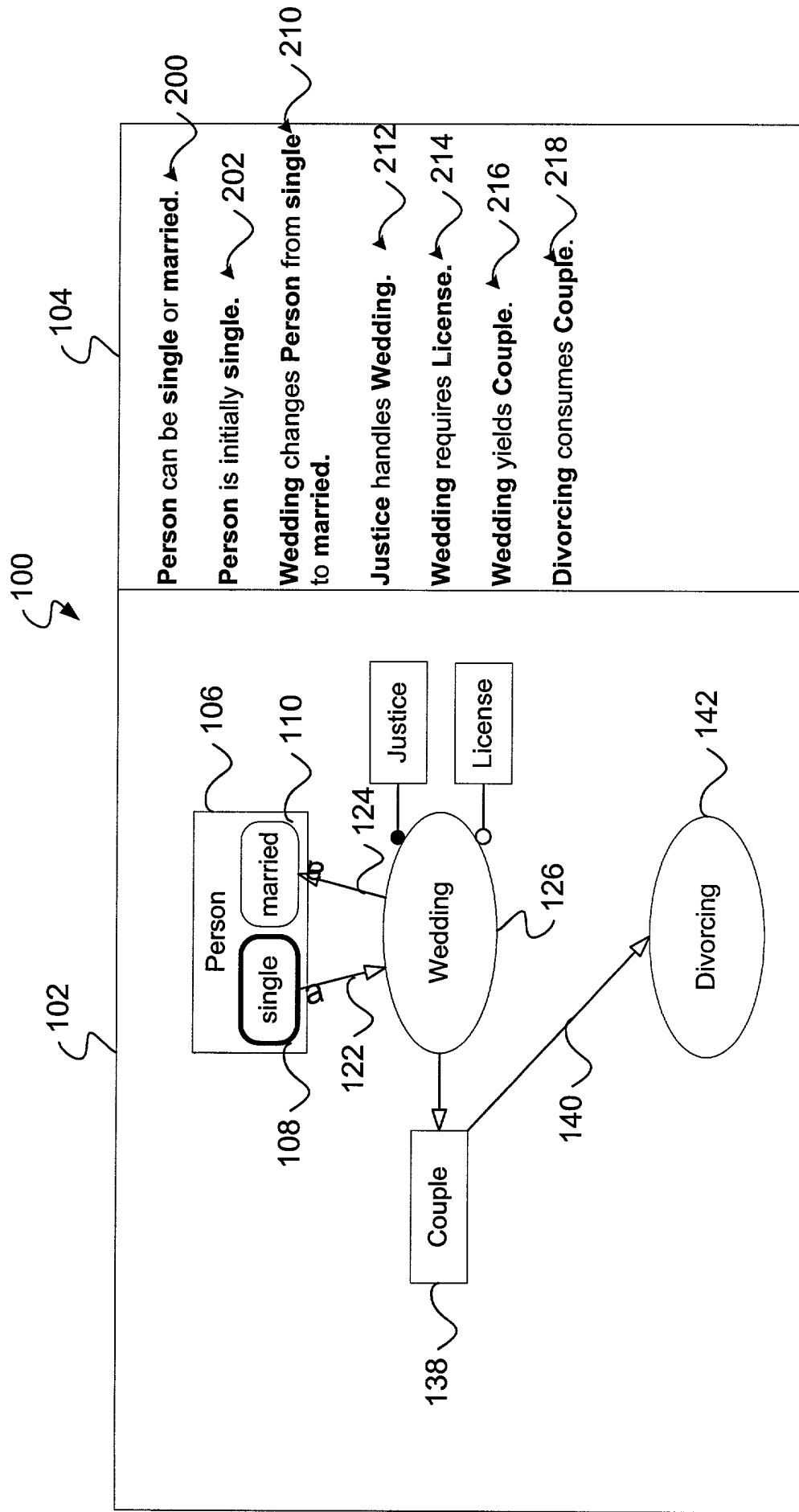


FIG. 6

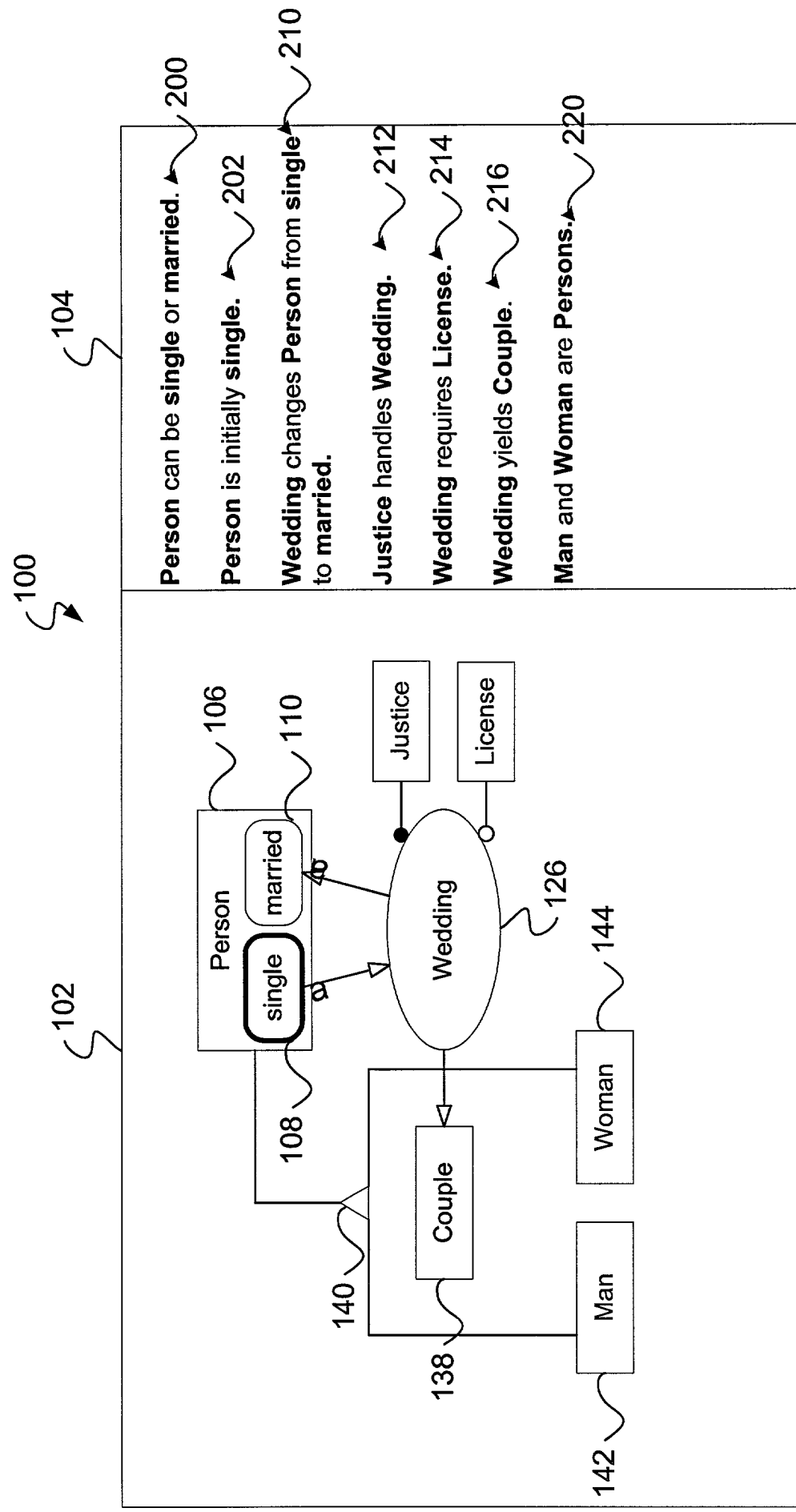


FIG. 7

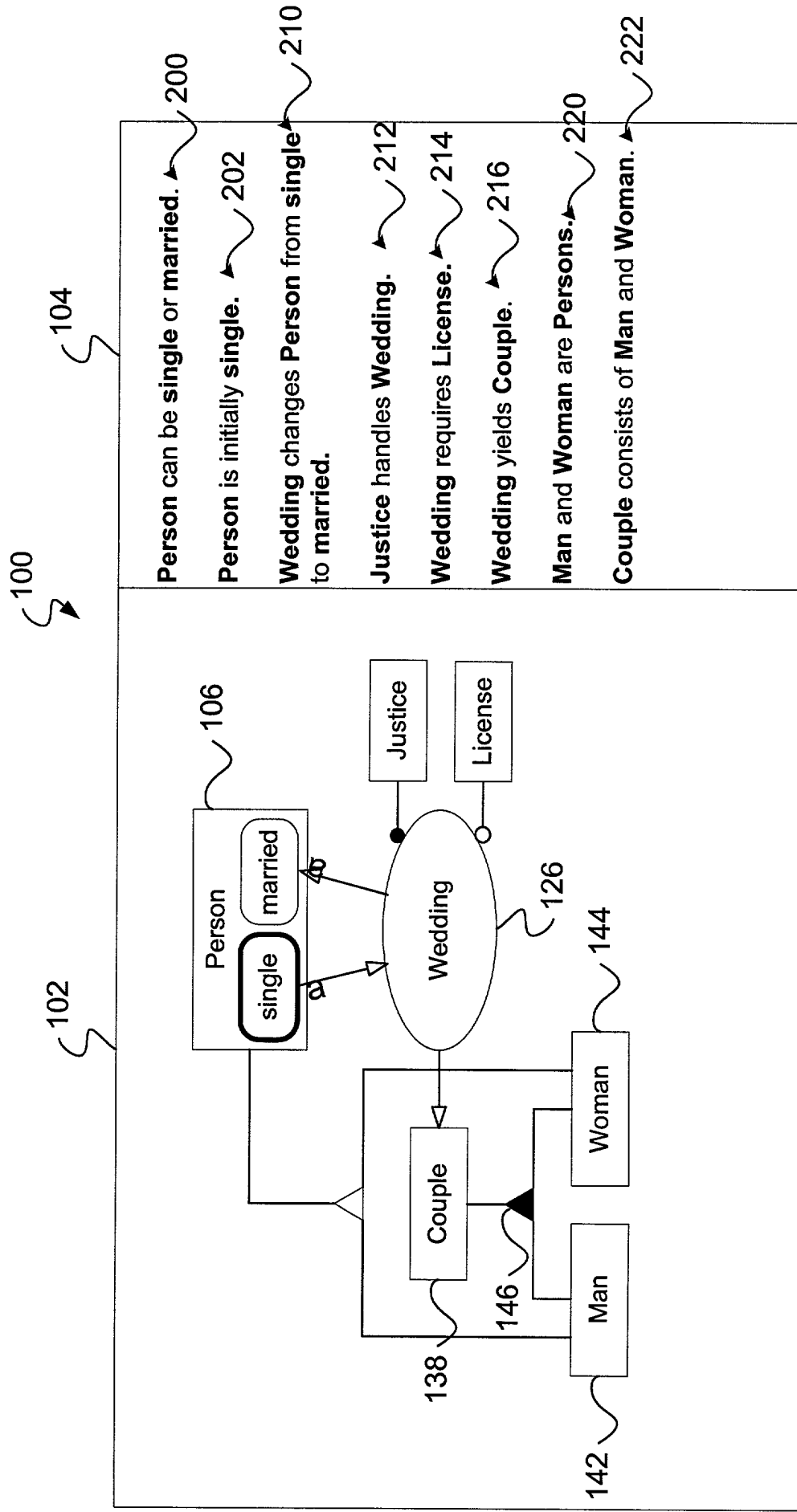


FIG. 8

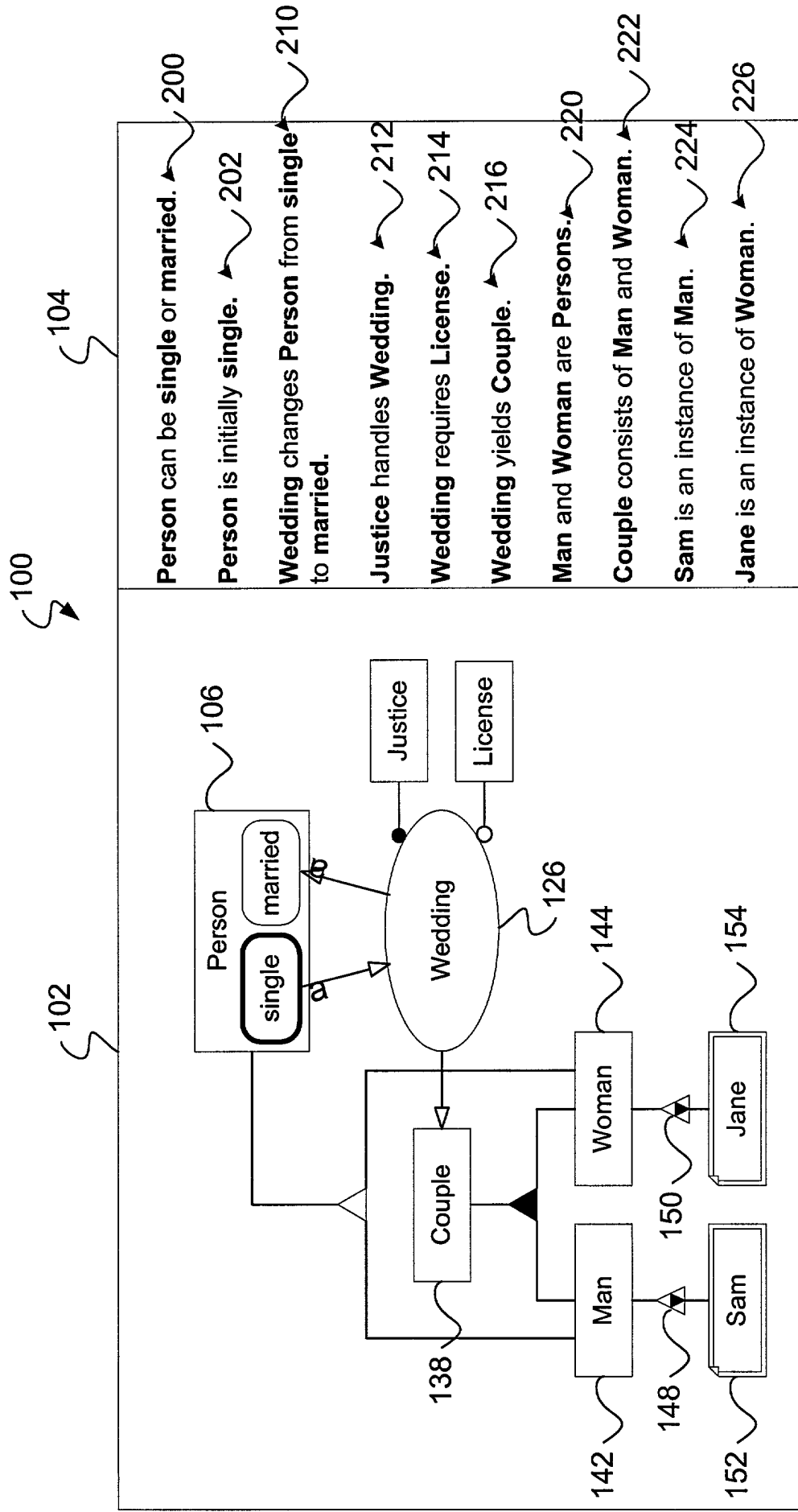


FIG. 9

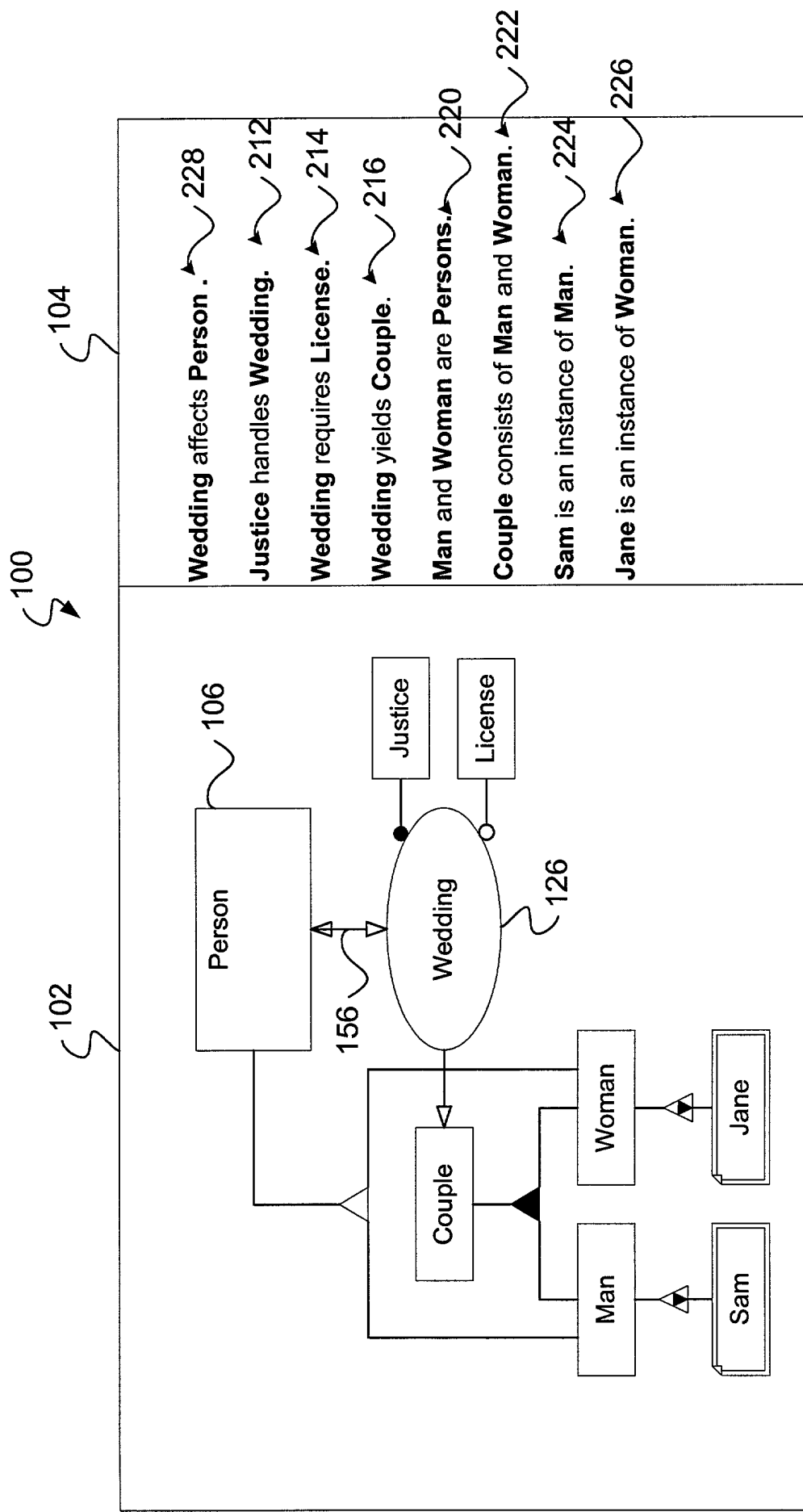


FIG. 10

FIG. 11 is a diagram of a system 100 for a wedding license. The system 100 includes a database 102, a user interface 104, and a processing unit 106. The database 102 stores information about couples 138, weddings 126, and licenses 134. The user interface 104 allows a user to interact with the system 100. The processing unit 106 is configured to receive input from the user interface 104, query the database 102, and provide output to the user interface 104. The system 100 is designed to manage wedding licenses and track the status of couples 138 and weddings 126.

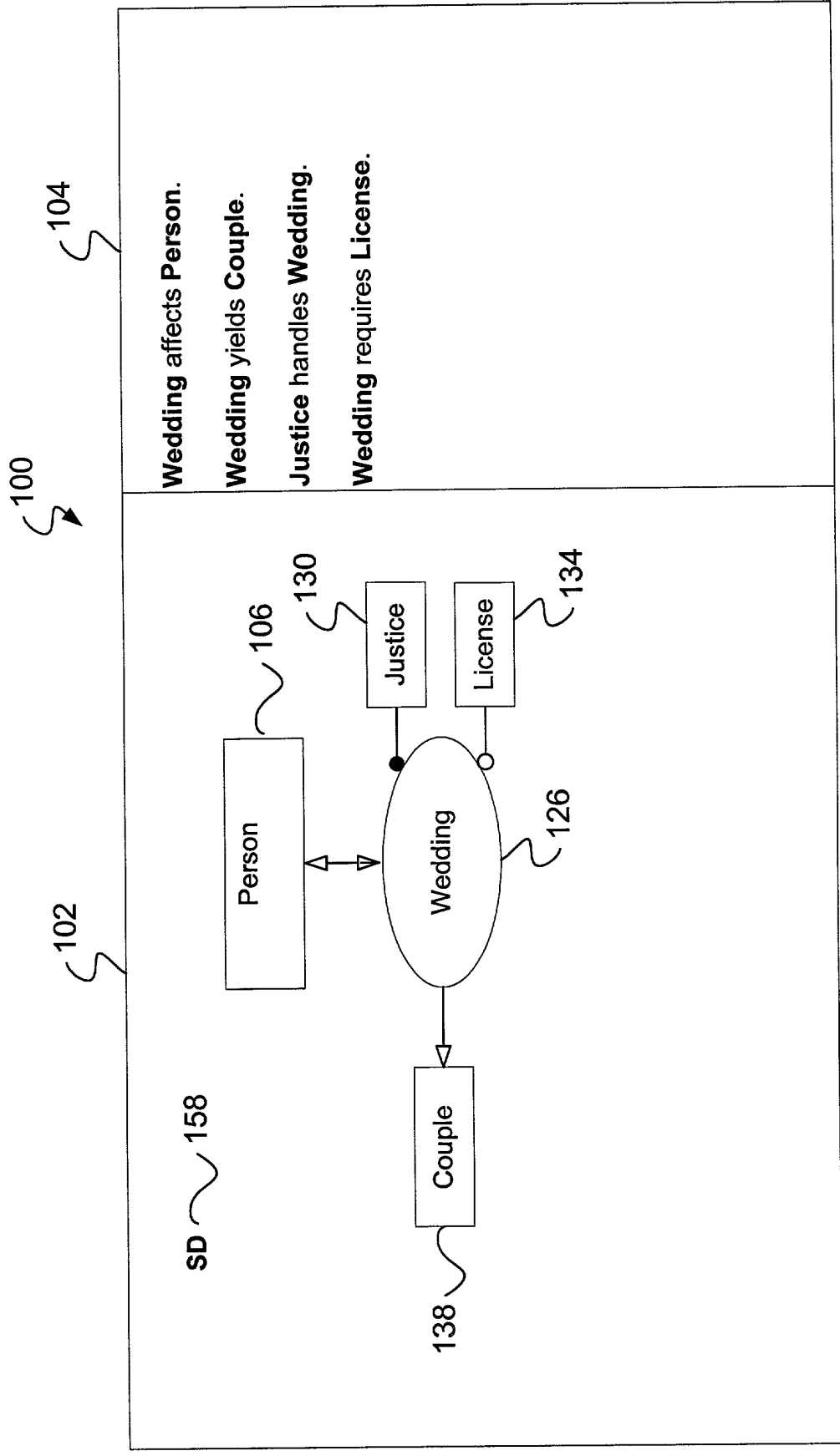


FIG. 11

100

102

104

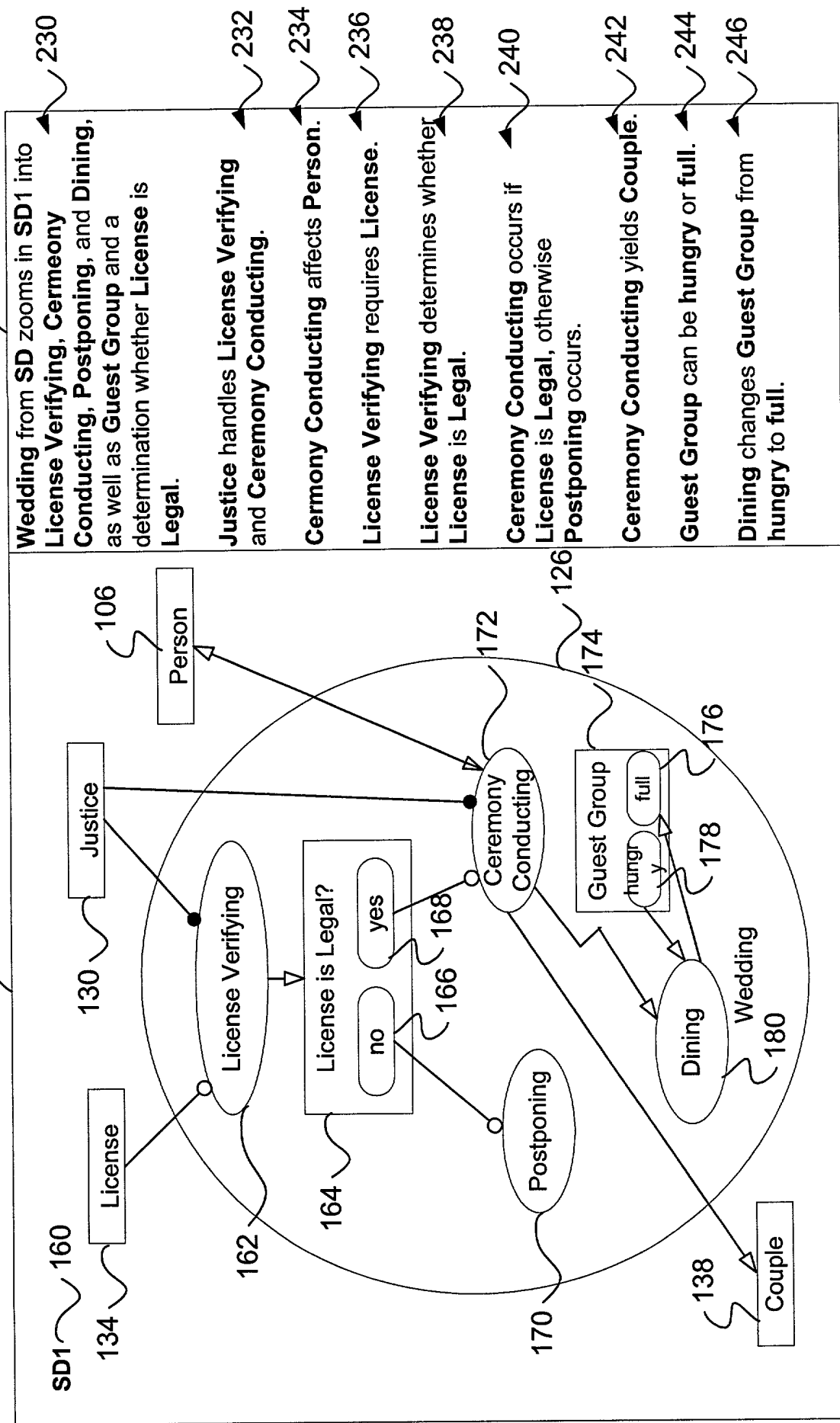


FIG. 12

FIG. 13 is a block diagram of a system architecture. The system includes a host 102, a storage device 150, and a storage device 180. The host 102 is connected to the storage device 150 via a connection 182. The storage device 150 is connected to the storage device 180 via a connection 184. The storage device 180 is connected to a storage device 190 via a connection 186. The storage device 190 is connected to a storage device 188 via a connection 188. The storage device 188 is connected to a storage device 190 via a connection 188.

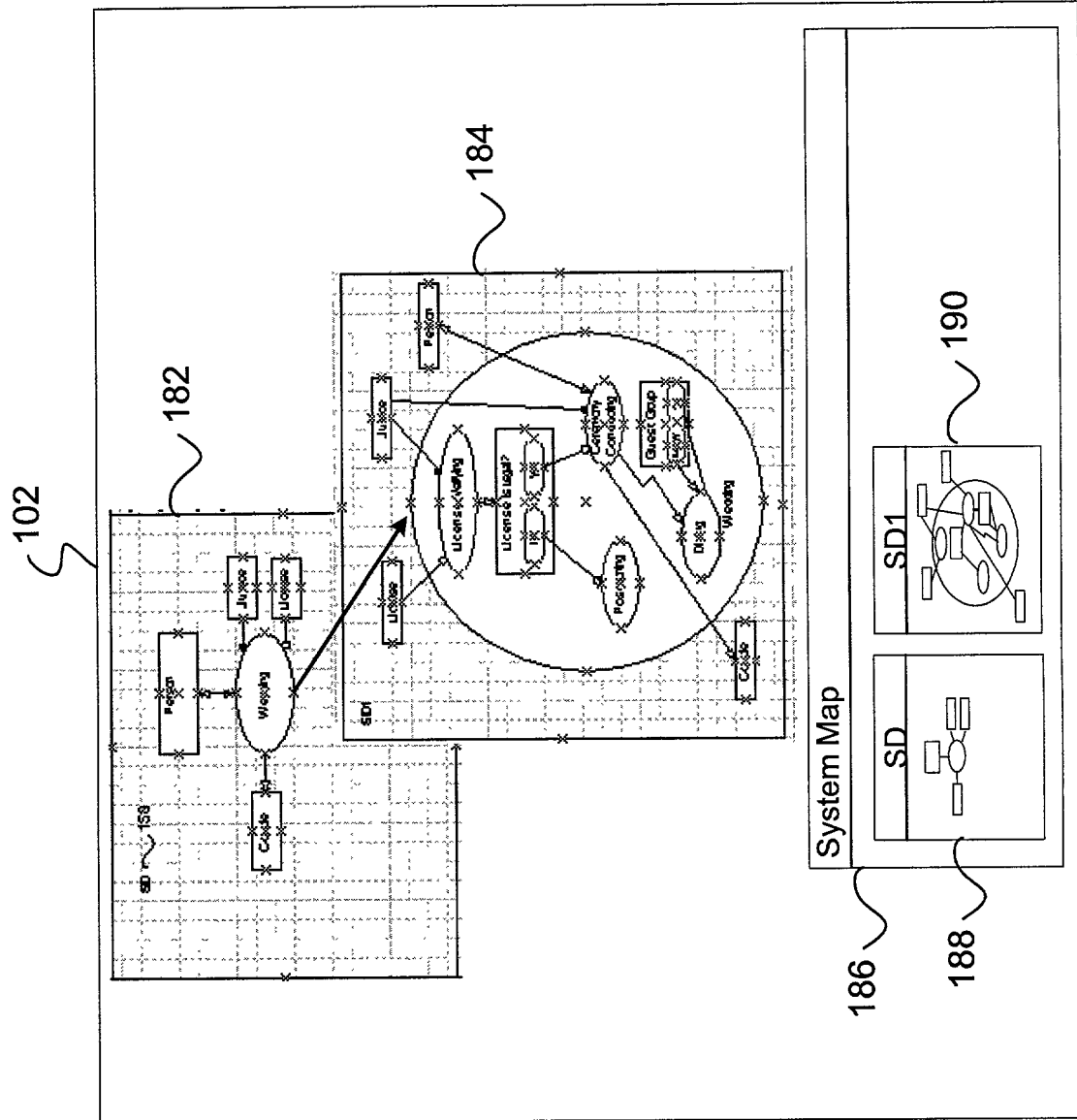


FIG. 13

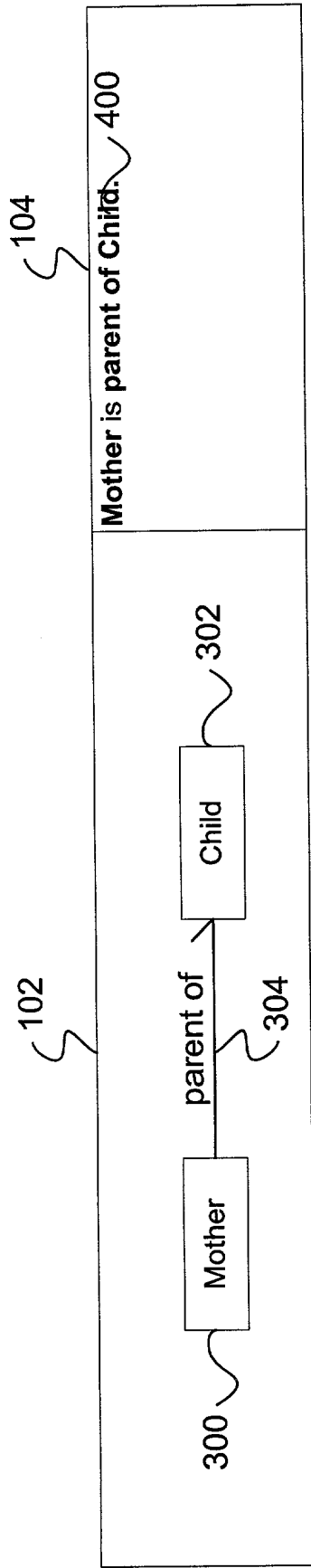


FIG. 14

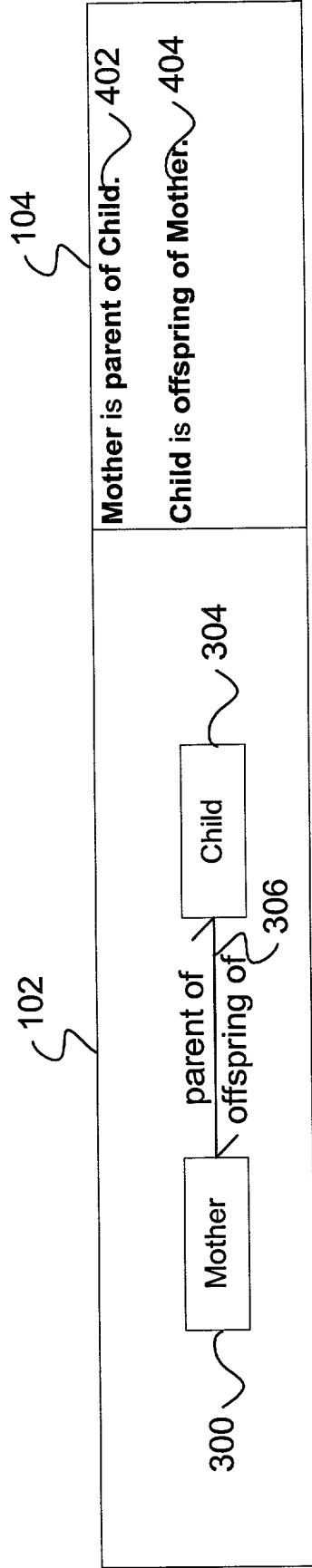


FIG. 15

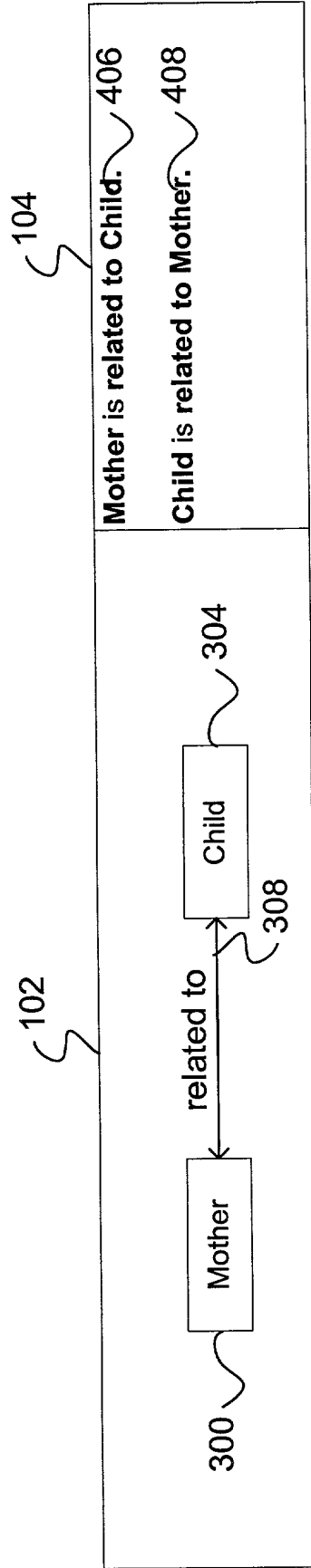


FIG. 16

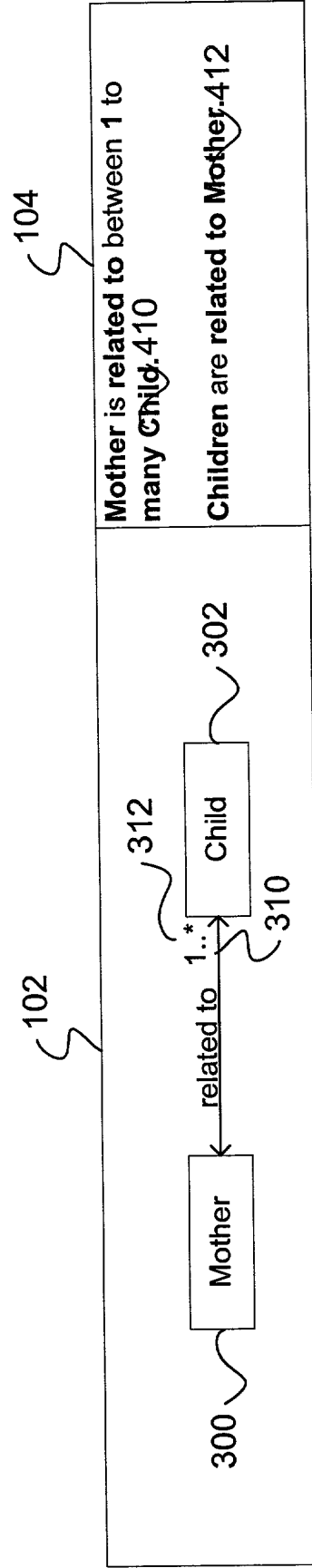


FIG. 17

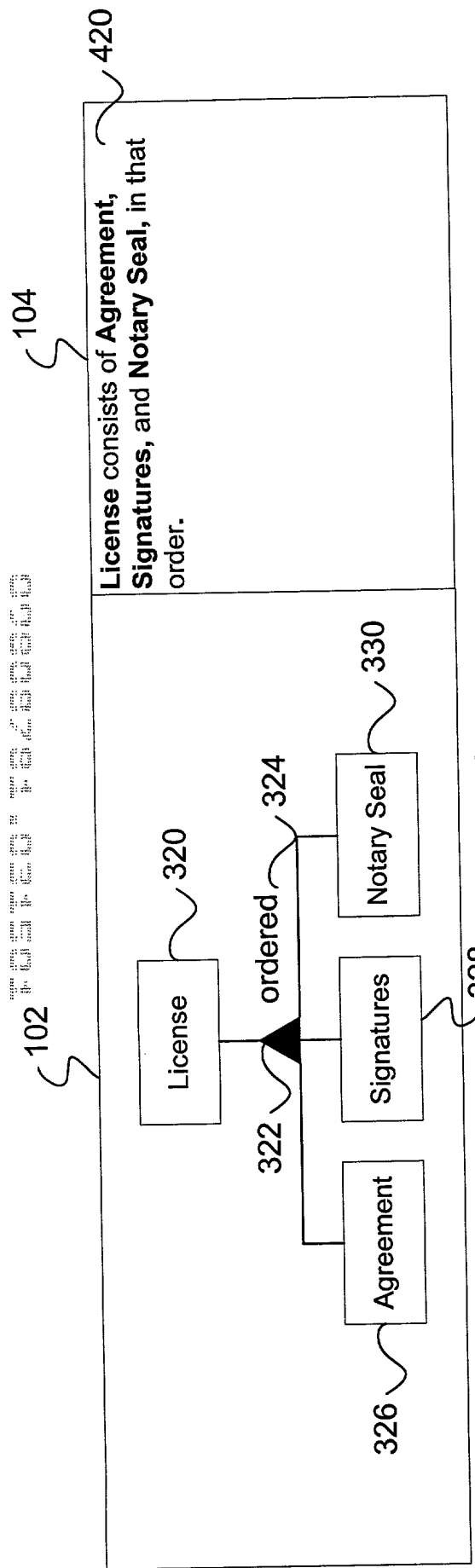


FIG. 18

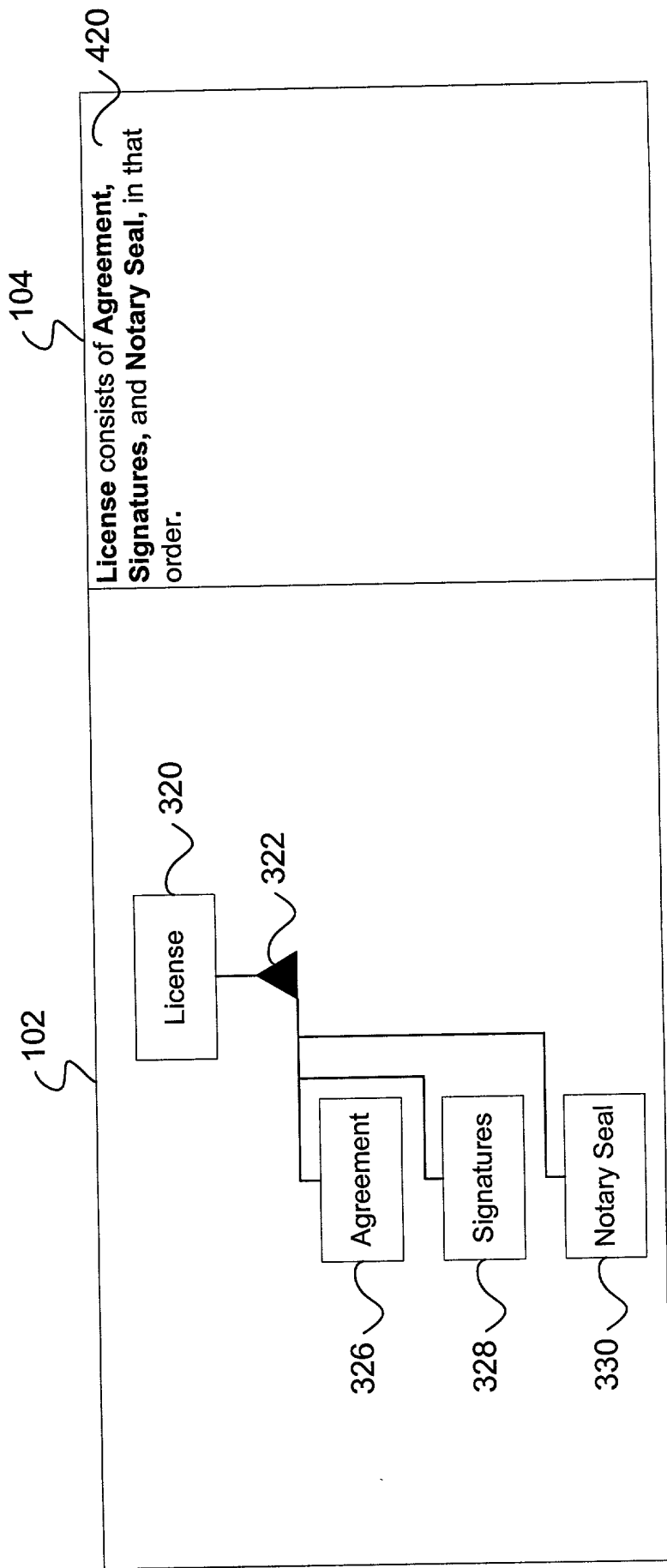


FIG. 19

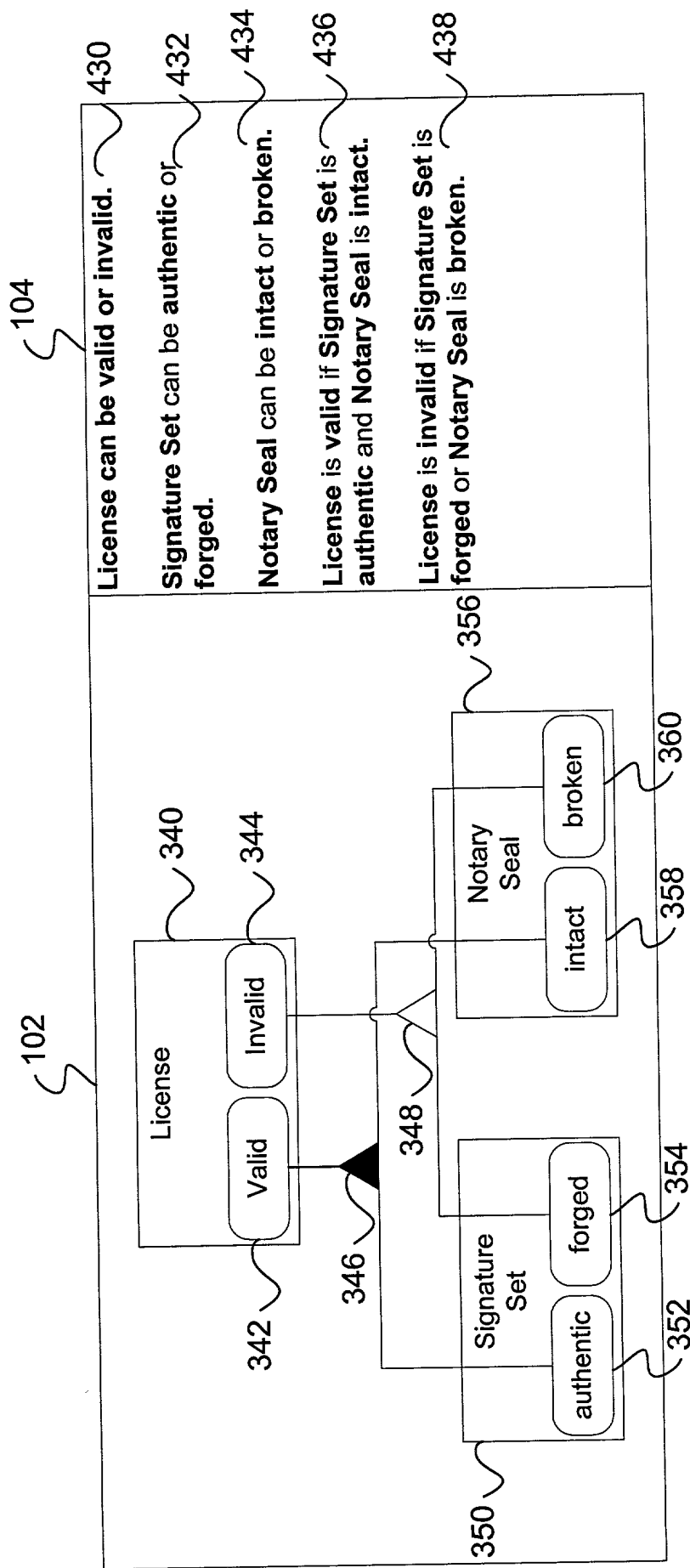


FIG. 20

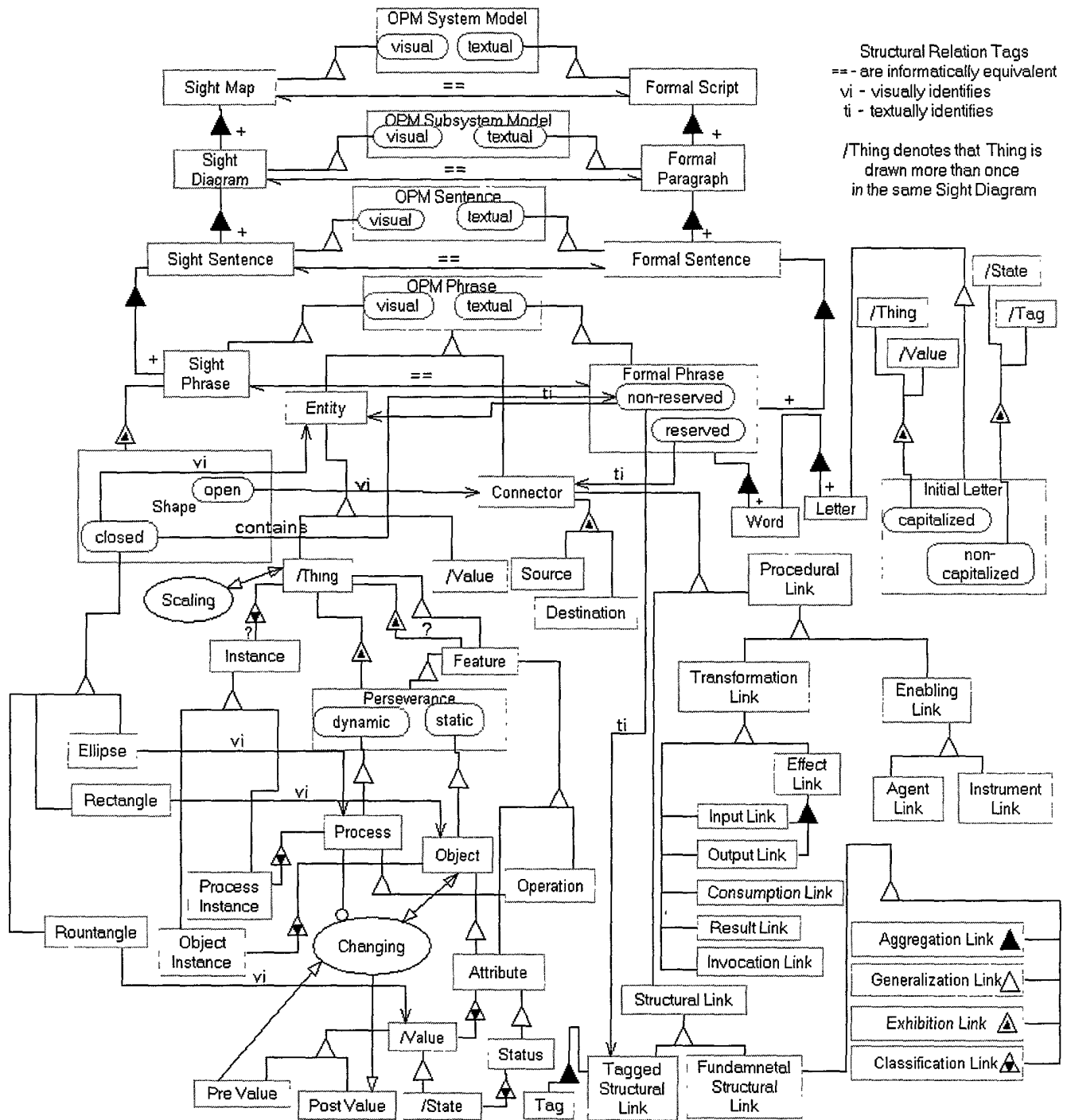


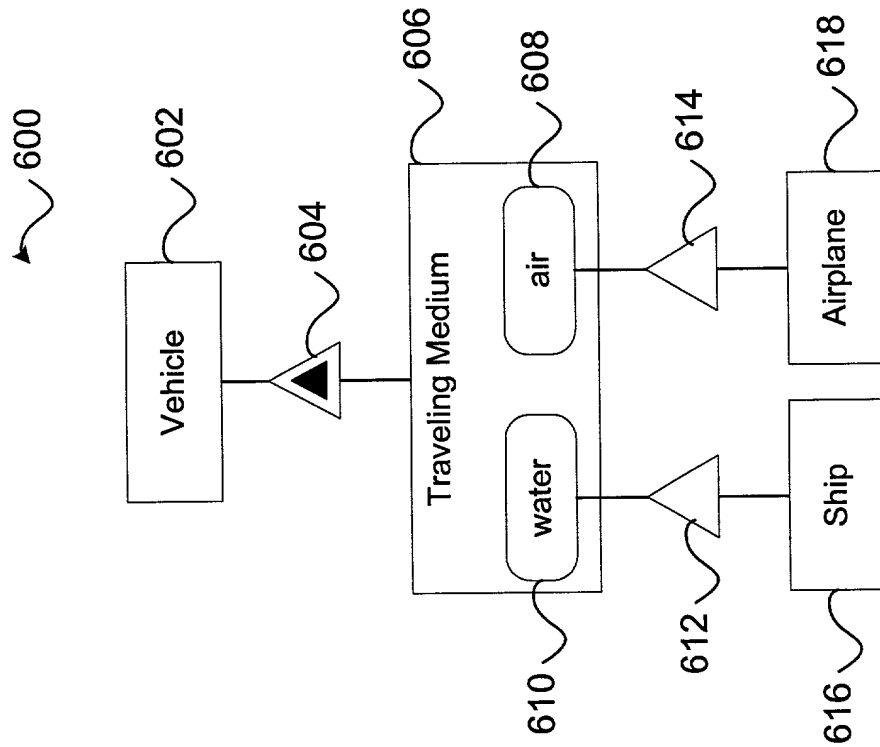
FIG. 21

OPM System Model can be visual or textual.
Sight Map is a visual OPM System Model.
Formal Script is a textual OPM System Model.
Sight Map and Formal Script are informatically equivalent.
Sight Map consists of one or more Sight Diagrams.
Formal Script consists of one or more Formal Paragraphs.
OPM Subsystem Model can be visual or textual.
Sight Diagram is a visual OPM Subsystem Model.
Formal Paragraph is a textual OPM Subsystem Model.
Sight Diagram and Formal Paragraph are informatically equivalent.
Sight Diagram consists of one or more Sight Sentences.
Formal Paragraph consists of one or more Formal Sentences.
OPM Sentence can be visual or textual.
Sight Sentence is a visual OPM Sentence.
Formal Sentence is a textual OPM Sentence.
Sight Sentence and Formal Sentence are informatically equivalent.
Formal Sentence consists of one or more Formal Phrases.
Sight Sentence consists of one or more Sight Phrases.
OPM Phrase can be visual or textual.
Sight Phrase is a visual OPM Phrase.
Formal Phrase is a textual OPM Phrase.
Sight Phrase and Formal Phrase are informatically equivalent.
Formal Phrase consists of one or more Words.
Word consists of one or more Letters.
Initial Letter is a Letter.
Initial Letter can be capitalized or non-capitalized.
Thing and Value exhibit capitalized Letter.
State and Tag exhibit non-capitalized Letter.
Formal Phrase can be non-reserved or reserved.
Sight Phrase exhibits Shape.
Shape can be open or closed.
Entity and Connector are OPM Phrases.
Closed Shape visually identifies Entity.
Non-reserved Formal Phrase textually identifies Entity.
Open Shape visually identifies Connector.

FIG. 22

Reserved Formal Phrase textually identifies **Connector**.
Closed Shape contains **non-reserved Formal Phrase**.
Thing and **Value** are **Entities**.
Scaling affects **Thing**.
Structural Link and **Procedural Link** are **Connectors**.
Transformation Link and **Enabling Link** are **Procedural Links**.
Connector exhibits **Source** and **Destination**.
Agent Link and **Instrument Link** are **Enabling Links**.
Effect Link, **Input Link**, **Output Link**, **Consumption Link**, **Result Link**, and
Invocation Link are **Transformation Links**.
Effect Link consists of **Input Link** and **Output Link**.
Fundamental Structural Link and **Tagged Structural Link** are **Structural Links**.
Non-reserved Formal Phrase textually identifies **Tagged Structural Link**.
Aggregation Link, **Generalization Link**, **Exhibition Link**, and **Classification Link**
are **Fundamental Structural Links**.
Thing instantiates **optional Instances**.
Thing exhibits **optional Features**.
Feature is a **Thing**.
Thing exhibits **Perseverance**.
Perseverance is a **Feature**.
Perseverance can be **static** or **dynamic**.
Object is a **Thing**, the **Perseverance** of which is **static**.
Process is a **Thing**, the **Perseverance** of which is **dynamic**.
Ellipse, **Rectangle**, and **Rountangle** are **closed Shapes**.
Rectangle visually identifies **Object**.
Ellipse visually identifies **Process**.
Attribute and **Operation** are **Features**.
Attribute is an **Object**.
Operation is a **Process**.
Attribute instantiates **Value**.
Rountangle visually identifies **Value**.
Status is an **Attribute**.
State instantiates **Status**.
State is a **Value**.
Changing affects **Object**.
Changing requires **Process**.
Changing consumes **Pre Value**.
Changing yields **Post Value**.

FIG. 23



Vehicle exhibits Traveling Medium. 622

Traveling Medium can be water or air. 624

Ship is a Vehicle, the Traveling Medium of which is water. 626

Airplane is a Vehicle, the Traveling Medium of which is air. 628

FIG. 24

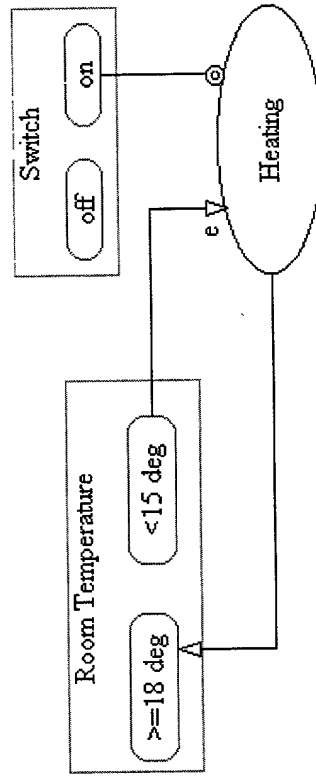


FIG. 25

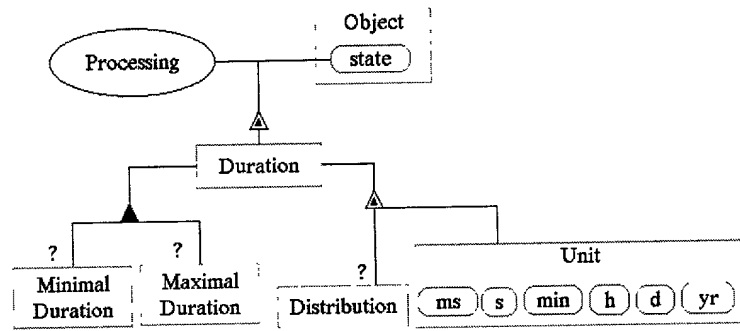


FIG 26

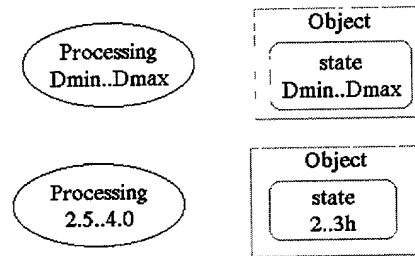
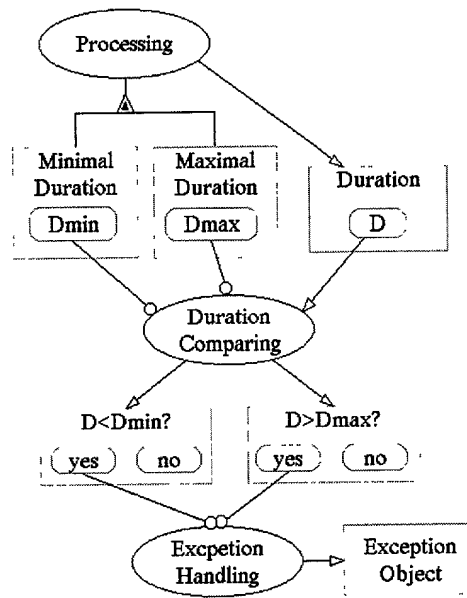


FIG 27



Processing exhibits **Minimal Duration** with value **Dmin**, and **Minimal Duration**, with value **Dmax**.

Processing yields **Duration** with value **D**.

Duration Comparing requires **Dmin** and **Dmax**.

Duration Comparing consumes **D**.

Duration Comparing consumes **D**.

Duration Comparing determines whether **D** is less than **Dmin**.

Duration Comparing determines whether **D** is more than **Dmax**.

Exception Handling occurs if either **D** is more than **Dmax** or **D** is less than **Dmin**.

Exception Handling yields **Exception Object**.

FIG 28

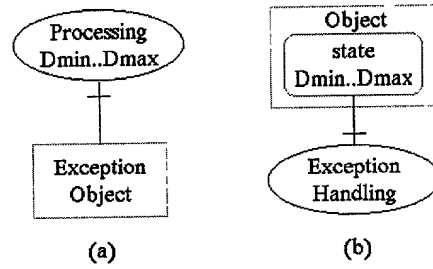


FIG 29

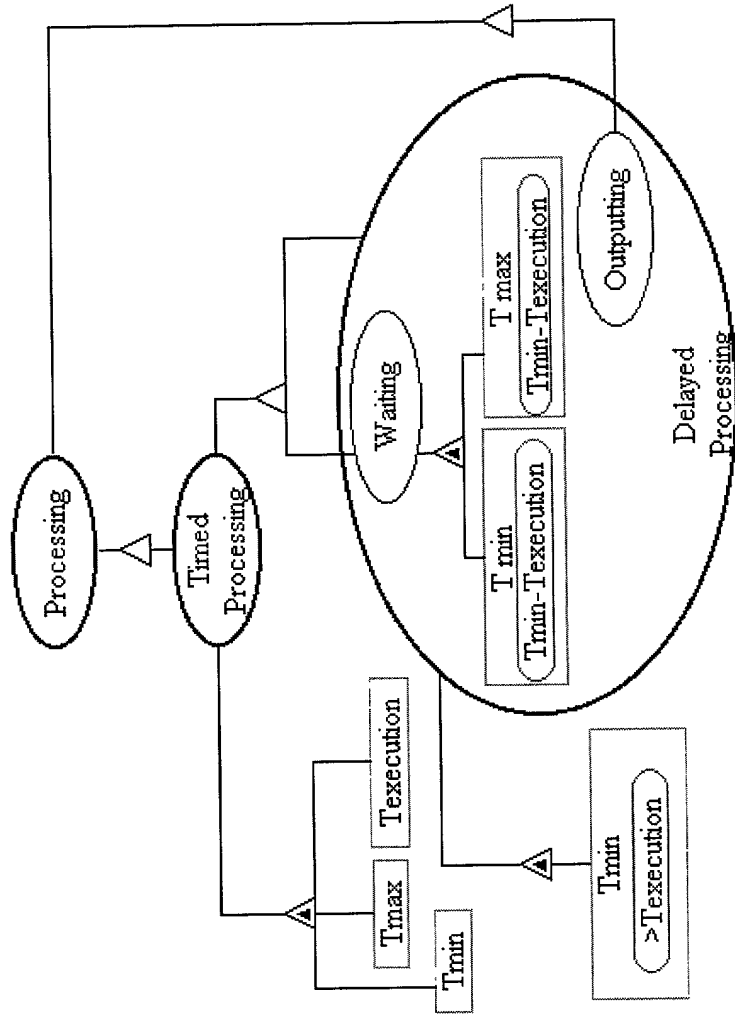
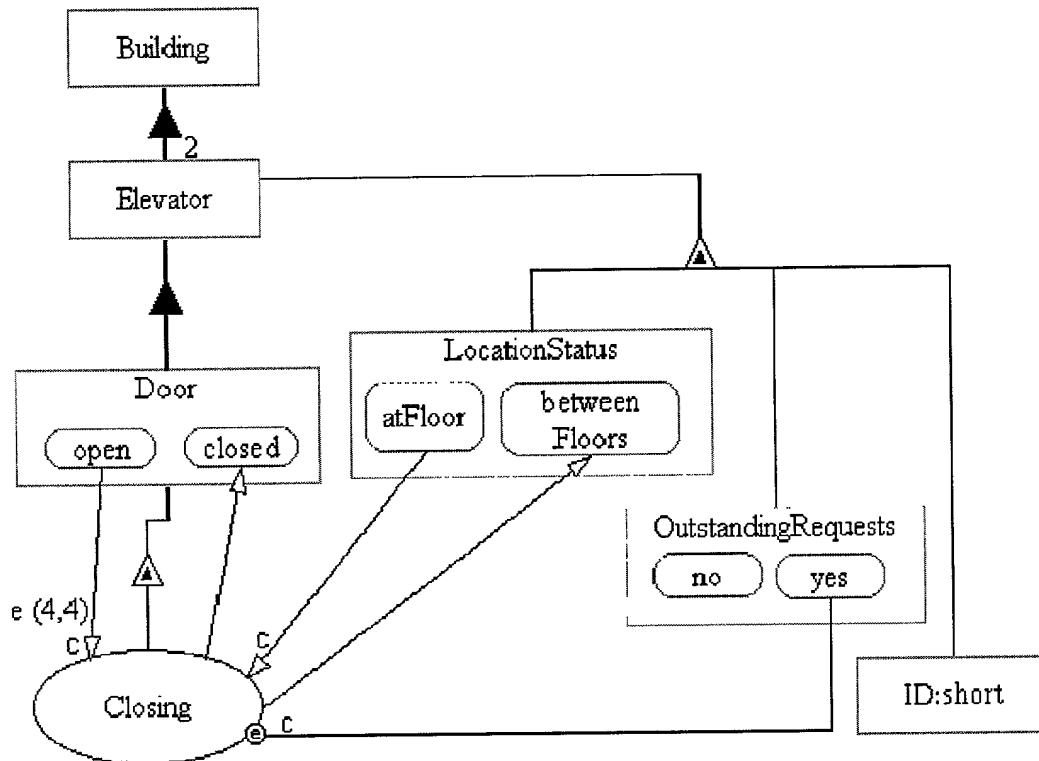


FIG. 30



Object **Building** consists of 2 **Elevators**.

Object **Elevator** features **ID**, **OutstandingRequests** and **LocationStatus**.

Object[simple] **ID:short** [key].

Object **OutstandingRequests** has states **no** and **yes**.

Object **LocationStatus** has states **atFloor** and **betweenFloors**.

Elevator consists of **Door**.

Object **Door** has states **open** and **closed**.

Door features process **Closing**.

Event **Door_open** [state-entrance] of **Door** of **Elevator** [**ID** in {1 to 2}] of **Building** triggers **Closing** of **Door** of **Elevator**[**ID**] of **Building** with a reaction time of (4,4).

Process **Closing** is guarded by "Door is open",

"OutstandingRequests of Elevator is yes" and

"LocationStatus of Elevator is atFloor".

Closing affects **Door** of **Elevator** from **open** to **closed** and

LocationStatus of **Elevator** from **atFloor** to **betweenFloors**.

FIG. 31

```

1 class Building : public Object {
2 private:
3   class Elevator : public Object {
4     private:
5       Building *f;
6       short ID;
7     public:
8       short get_ID(void);
9       void set_ID(short t);
10    private:
11      class Door : public Object {
12        private:
13          Elevator *f;
14          State* closed;    // same for State open: ...
15        public:
16          void set_closed(BOOLEAN enter, short Invoker);
17          void enter_closed(short Invoker);
18          void enter_closed(void);
19          enum door get_Door(void);
20          friend Elevator;
21        private:
22          class Closing : public Process {
23            private:
24              Door *f;
25              void Execute(short Invoker, short Triggering_Event);
26            public:
27              Closing(void);
28              BOOLEAN Trigger(short Invoker, short
29                Triggering_event);
30              friend Door;
31            };
32          public:
33            Closing the_Closing;
34            friend Closing;
35            Door(void);
36          };
37        public:
38          Door the_Door;
39          // LocationStatus and OutstandingRequests are defined in
40          // a way similar to Door
41          ...
42          Elevator(void);
43        };
44      public:
45        Elevator the_Elevator[2];
46        Building(void);
47      };
48    extern Building the_Building;

```

Elevator features
simple object ID

Elevator
consists of Door

Door features
Process Closing

FIG. 32

```

BOOLEAN Building::Elevator::Door::Closing::Trigger(short Invoker, short
Triggering_event)
{
    BOOLEAN result = TRUE;

    if (f->get_Door()==OPEN)
        result = TRUE; // if the Elevator Door is OPEN then
        // proceed to check if there are
        // OutstandingRequests for the Elevator
    else
        result = FALSE;
    if (result == TRUE)
    {
        if
            (f->f->the_OutstandingRequests.get_OutstandingRequests()==YES)
                result = TRUE; // if there are OutstandingRequests then
                // proceed to check if Elevator is ATFLLOOR
        else
            result = FALSE;
    }
    if (result == TRUE)
    {
        if (f->f->the_LocationStatus.get_LocationStatus()==ATFLLOOR)
            result = TRUE; // if if Elevator is ATFLLOOR then
            // proceed to open the Elevator Door
        else
            result = FALSE;
    }
    if (result == TRUE)
    {
        Execute(Invoker, Triggering_event); // open the Door
        return TRUE;
    }
    else
        return FALSE;
    }
}

```

FIG. 33

```

void _EventProcessTable::TriggerAllProcesses(Event *e)
{
    Process *process_ptr;

    switch(e->get_id())        // get id of triggering_Event
        // (e.g., door_open)
    {
        case dooropen: switch (e->get_trig_obj()){
                        // get id of triggering object
                        case elevator1:
                            process_ptr =
                                &the_Building.the_Elevator[0].the_Door.the_Closing;
                            break;
                        case elevator2:
                            process_ptr =
                                &the_Building.the_Elevator[1].the_Door.the_Closing;
                            break;
                    }
        TryToTrigger(e,process_ptr, 0,4,0,0,0,4,0,0,
                    e->get_trig_obj(), e->get_id());
        break;
    }
}

```

FIG. 34

```

void Building::Elevator::Door::Closing::Execute(short Invoker,
short Triggering_event)
{
    Process::Execute(Invoker, Triggering_Event);
    f->f->the_Door.enter_closed(Invoker);
    f->f->the_LocationStatus.enter_betweenFloors();
}

```

FIG. 35

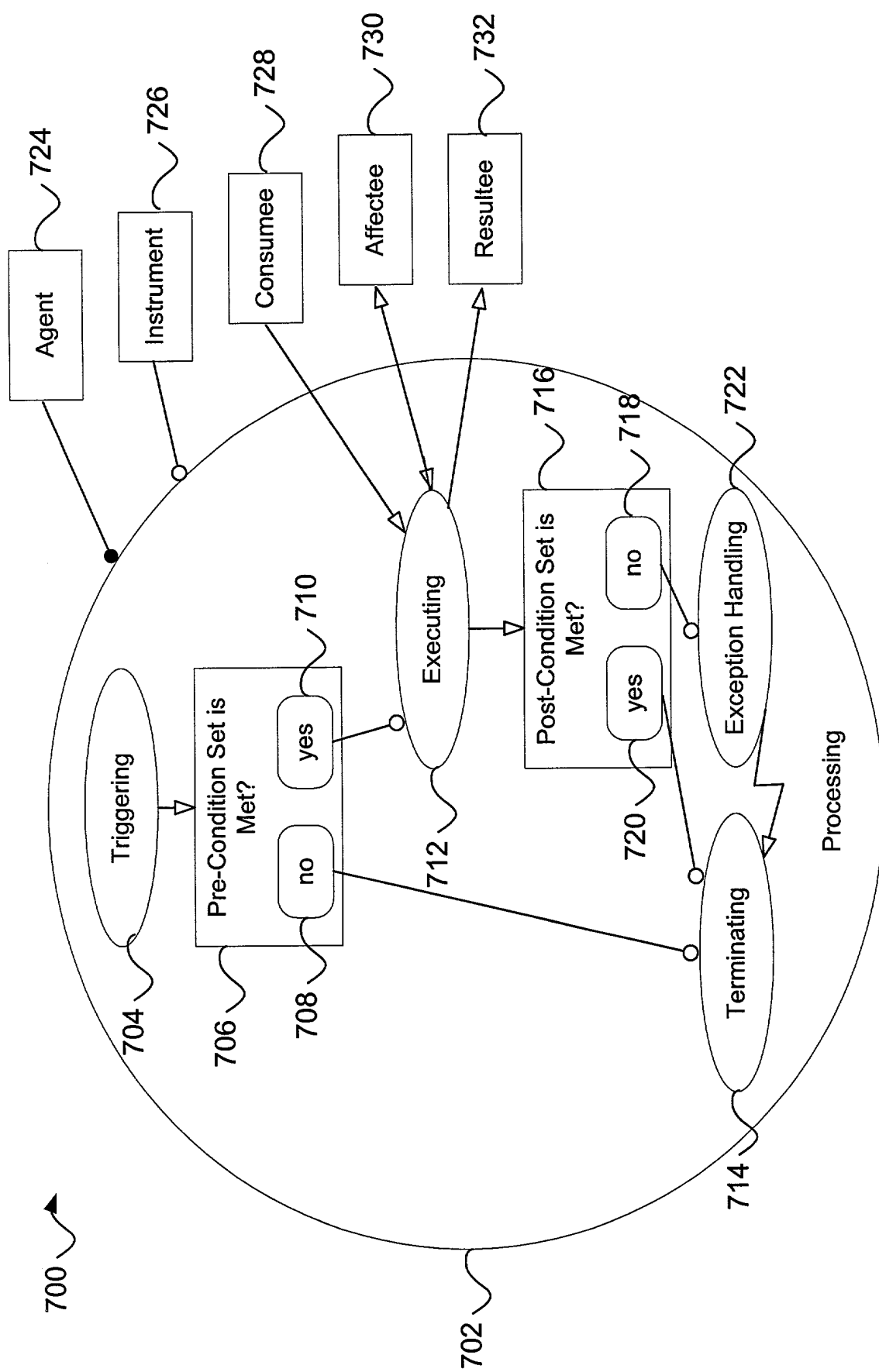


FIG. 36

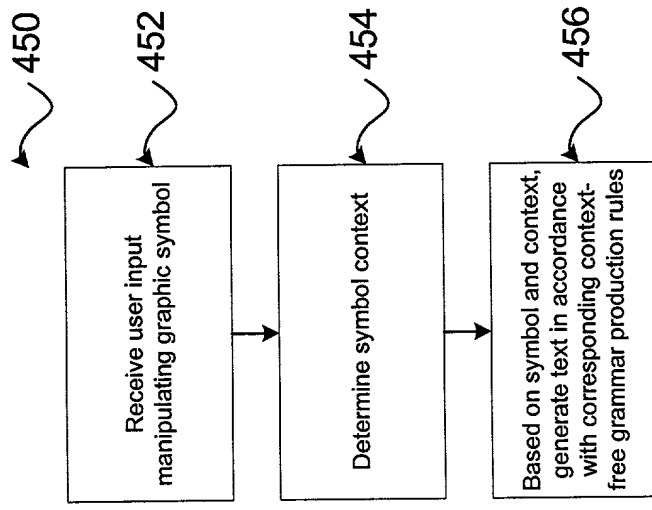


FIG. 37

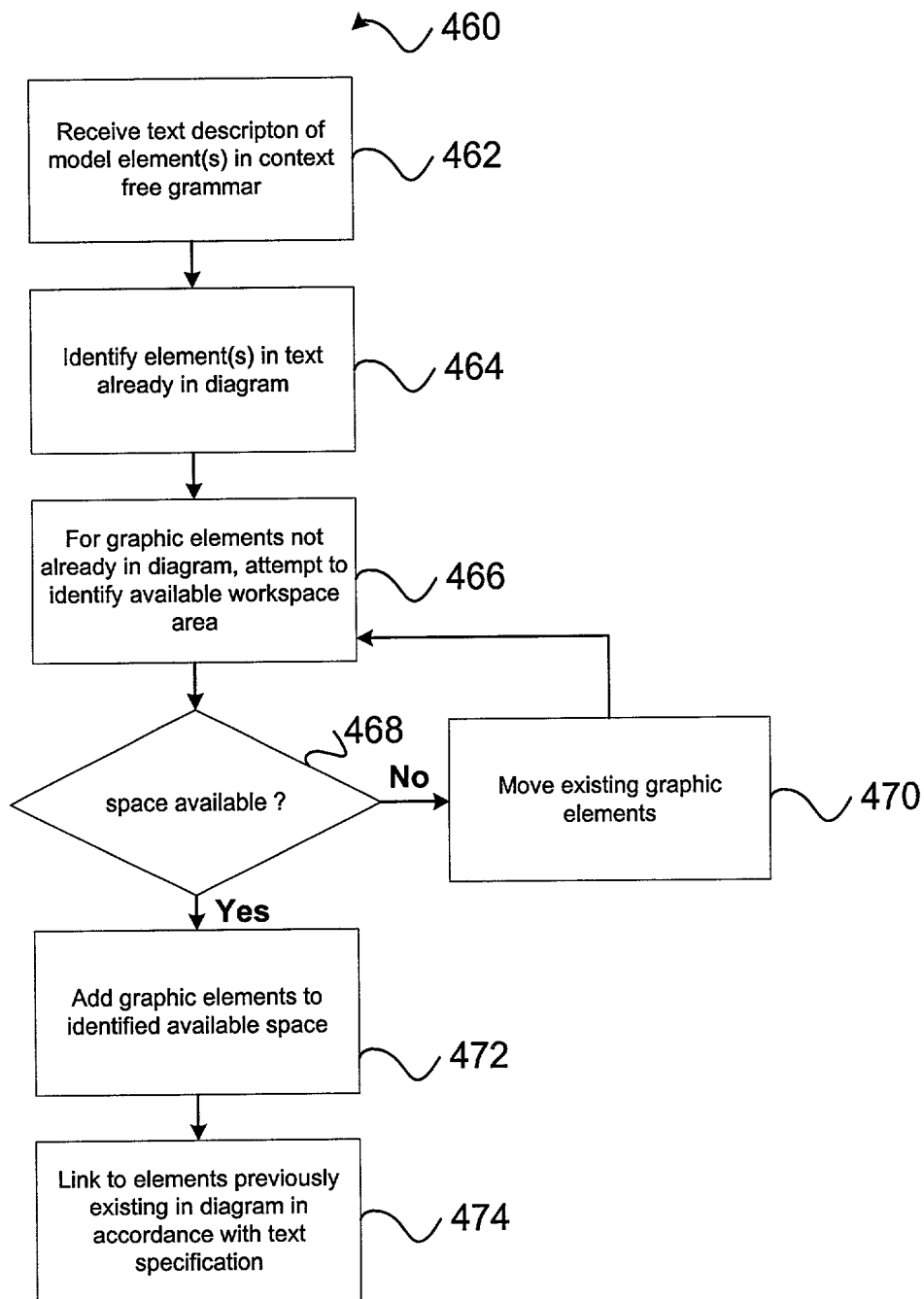
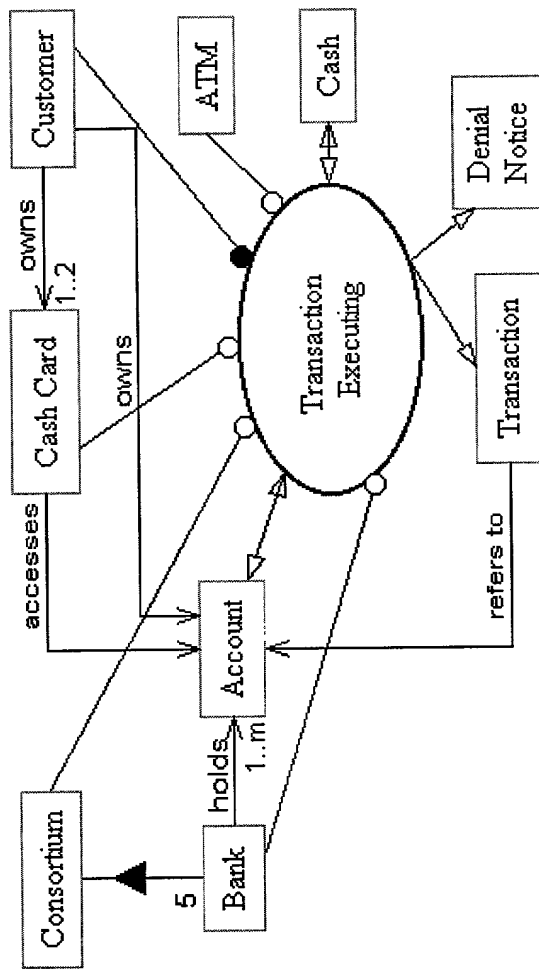


FIG. 38

100

102

104



Consortium consists of 5 Banks.
 Bank holds many Account.
 Customer owns Account.
 Customer owns between 1 to 2 Cash Cards.
 Cash Card accesses Account.
 Customer handles Transaction Executing.
 Transaction Executing requires ATM, Cash Card, Consortium and Bank.
 Transaction Executing affects Account and Cash.
 Transaction Executing yields either Transaction or Denial Notice.

FIG. 39

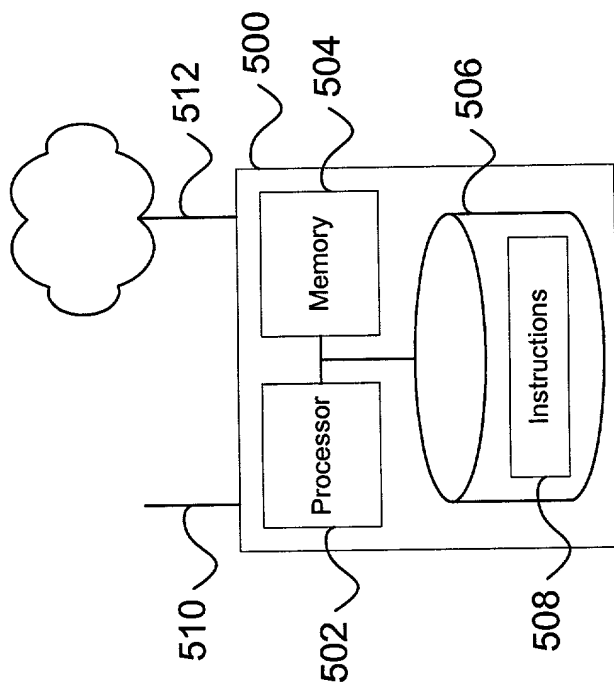


FIG. 41